mathematics

MDPI

Article

A Mixed-Integer Program for Drawing Orthogonal Hyperedges in a Hierarchical Hypergraph

Gregory Fridman [1,*], Yuri Vasiliev [1], Vlada Puhkalo [1] and Vladimir Ryzhov [2]

[1] Department of Applied Mathematics and Mathematical Methods in Economics, Saint Petersburg State University of Economics, Griboedov Canal Emb., 30-32, 191023 St. Petersburg, Russia; vas_yu_m@mail.ru (Y.V.); suzdareva@gmail.com (V.P.)

[2] Department of Applied Mathematics and Mathematical Modeling, Saint Petersburg State Marine Technical University, Lotsmanskaya Ulitsa, 3, 190121 St. Petersburg, Russia; varyzhov@smtu.ru

* Correspondence: grifri@finec.ru

Abstract: This paper presents a new formulation and solution of a mixed-integer program for the hierarchical orthogonal hypergraph drawing problem, and the number of hyperedge crossings is minimized. The novel feature of the model is in combining several stages of the Sugiyama framework for graph drawing: vertex ordering, the assignment of vertices' x-coordinates, and orthogonal hyperedge routing. The hyperedges of a hypergraph are assumed to be multi-source and multi-target, and vertices are depicted as rectangles with ports on their top and bottom sides. Such hypergraphs are used in data-flow diagrams and in a scheme of cooperation. The numerical results demonstrate the correctness and effectiveness of the proposed approach compared to mathematical heuristics. For instance, the proposed exact approach yields a 67.3% reduction of the number of crossings compared to that obtained by using a mathematical heuristic for a dataset of non-planar graphs.

Keywords: directed acyclic hierarchical graph; graph drawing; hypergraph; orthogonal hyperedge routing; branch-and-bound method; mathematical heuristic; financial flow

MSC: 05C65; 68R10; 90C90

check for updates

Citation: Fridman, G.; Vasiliev, Y.; Puhkalo, V.; Ryzhov, V. A Mixed-Integer Program for Drawing Orthogonal Hyperedges in a Hierarchical Hypergraph. *Mathematics* **2022**, *10*, 689. https://doi.org/10.3390/math10050689

Academic Editor: Irina Cristea

Received: 20 December 2021
Accepted: 21 February 2022
Published: 23 February 2022

1. Introduction

A new optimization model for the layout problem of a hierarchical orthogonal hypergraph is discussed in this paper. The outputs of the model are node coordinates and hyperedge routing. The number of hyperedge crossings is minimized for better visibility in the graph layout, and some aesthetic metrics are optimized as well [1,2].

Every orthogonal hyperedge e is drawn with a set of upper vertical segments $sU(e)$, a horizontal segment $sH(e)$, and a set of lower vertical segments $sL(e)$. It is assumed in this paper that all hyperedges are directed downwards. Each upper vertical segment of $sU(e)$ has the same length and starts in one of the ports of the source node; the number of ports and their coordinates are known for each node. The horizontal segment $sH(e)$ is placed on the ordinate of the lower end of the vertical segments of $sU(e)$; $sH(e)$ and each vertical segment of $sU(e)$ have one conjunction point. The vertical segments of $sL(e)$ start on the $sH(e)$ segment and arrive at one of the ports of their target nodes. Figure 1 shows an example of the layering of a hierarchical hypergraph with orthogonal hyperedges according to the described rules.

A well-known approach to hyperedge visualization is the replacement of each hyperedge with a set of edges with one dummy node, followed by the use of traditional layout techniques for graphs; then, the hypergraph is restored from the graph layout obtained in the previous step [3,4]. Therefore, this problem is closely related to hierarchical (i.e., directed acyclic) graph drawing, and methods based on the Sugiyama approach [5] will be applied.

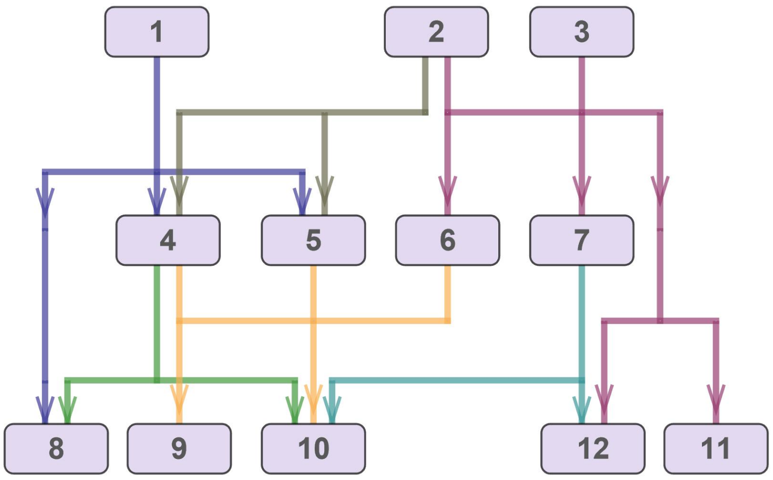

Figure 1. An example of the layering of a hierarchical hypergraph with orthogonal hyperedges according to the accepted rules.

To make this algorithm applicable to the hypergraph layout problem, Sander [6] first proposed reordering the nodes on each layer to minimize the number of edge crossings, and then calculated the preliminary coordinates for the nodes and segments to avoid hyperedge crossings, finally obtaining the coordinates that rendered a balanced drawing. In [7], an exact formulation of the mathematical problem of hyperedge routing was proposed, and it satisfied the constraints discussed above. The final step of the hypergraph drawing algorithm is assigning each upper vertical segment to a port of its source node and each lower vertical segment to a port of its target node.

Computing the optimal order of vertices for all layers of a graph is a highly time-consuming procedure. That is why it is reasonable to reorder vertices in each layer separately to minimize the number of edge crossings related to one of adjacent layers. A heuristic proposed in [8] was improved in further work; see [9].

Plenty of exact and heuristic algorithms have been presented for the minimization of edge crossings by reordering nodes on layers; see, e.g., [8,10]. However, application of these methods for a hypergraph might not provide acceptable results. It is known that decreasing the number of edge crossings does not ensure the same for the corresponding hypergraph [11]; see, for instance, Figures 2 and 3, which present examples of single-source one-port hypergraphs.

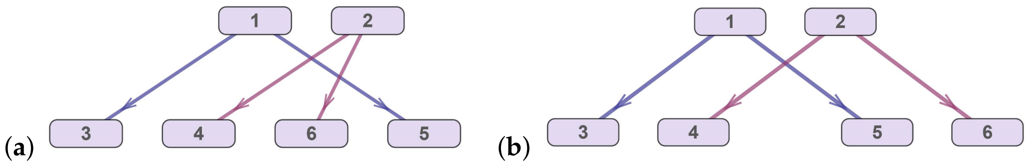

Figure 2. Reordering nodes on the lower layer of a graph leads to a decrease in the number of edge crossings: two crossings for the layout (**a**) and one crossing for the layout (**b**).

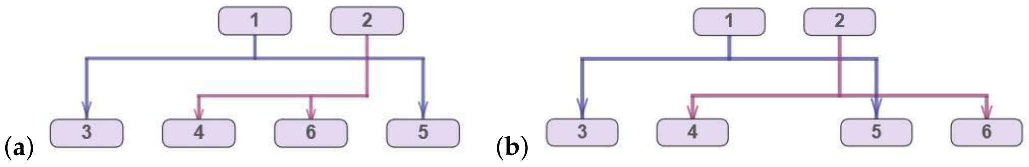

Figure 3. The same reordering nodes on the lower layer of the corresponding hypergraph leads to an increase in the number of hyperedge crossings: one crossing for the layout (**a**) and two crossings for the layout (**b**).

An additional optimization problem arises if one takes ports for the source and target nodes into account because each vertical segment of a hyperedge must be assigned to a unique port. This problem should be considered as a part of a hierarchical hypergraph drawing problem, together with the reordering of nodes and horizontal segments. Figures 4 and 5 show three layouts of the same hypergraph: without ports (optimal layout), with the assignment of ports as a result of a post-processing procedure, and, finally, with the optimal assignment of ports.

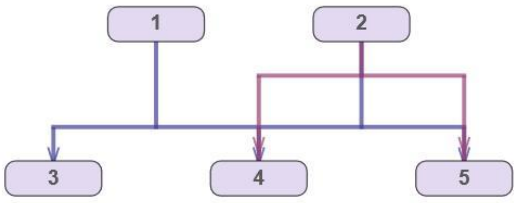

Figure 4. Optimal layout of a hypergraph; no ports allowed.

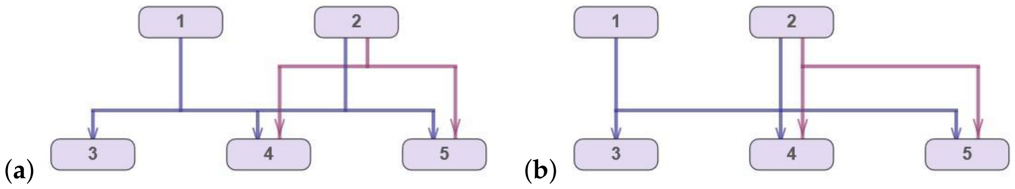

Figure 5. Layout of a hypergraph with ports for the source and target nodes: (**a**) non-optimal layout; (**b**) optimal layout with ports.

The reordering heuristic by Eschbach et al. [12] only gives a local minimum of the number of hyperedge crossings. Spönemann et al. [11] proposed an algorithm for quite accurate (but not exact) counting of hyperedge crossings depending on the order of nodes without computing the hyperedge routing.

A hyperedge drawing is applied in a variety of practical areas: from cooperation scheme drawing and information and financial flow visualization to schematics of displays of automotive communication networks and VLSI design. Interesting applications of the hypergraph layout problem can be found in the development of effective routing protocols for IoT applications and in the development of energy-balanced routing protocols for terrestrial/underwater wireless sensor networks, as well as in the assessment of their safety.

The information system for the national investment projects that started in Russia in 2015 is an example of graphical data analysis [7]. One of the goals of this information system is to control all financial transactions and contracts related to large-scale national projects, in which hundreds of companies are involved. The embedded software visualizes companies as nodes of a hierarchical hypergraph and their business links and financial transactions as directed hyperedges. This visualization scheme enables one to highlight potential risks and to estimate the status of a project under consideration. It is clear that this visualization is only helpful as an analytical tool if it is easy to understand. It is more to the point that the hypergraph density is high.

The neighboring area of application is financial flow visualization for corporations with large numbers of branch offices and commercial subsidiaries. The display of the flow of information within or between universities can be discussed as an application as well.

There are some novel approaches to nanostructures and material science with application of graph theory [13,14], where there is a large variety of applications connected with hierarchical hypergraphs with orthogonal hyperedges.

2. Requirements for Hyperedge Drawing

In mathematical terms, a directed k-layering hypergraph $H = (V, E_H, \lambda)$ contains a set V of nodes and a set E_H of hyperedges. A hyperedge $e = (S, T)$ has source nodes $S \subset V$ and target nodes $T \subset V$. A layering function λ is introduced in order to partition the set of vertices V into a k finite subsets (layers) $V_1, V_2, \ldots, V_k$. The function λ assigns a positive integer to each vertex $v \in V$ (layer), $1 \leq \lambda(v) \leq k$.

We assume the following set of requirements for orthogonal hyperedge routing, which is complementary to that for a layered graph drawing [10]:

1. Each hyperedge e consists of three elements: upper vertical segments $sU(e)$, horizontal segments $sH(e)$, and lower vertical segments $sL(e)$;
2. Each hyperedge is incident to one or more source vertices and adjacent (target) vertices of the lower layers;
3. Vertices are represented by rectangles of a fixed size with a set of ports on their upper and lower sides. In fact, every port is a common point of a vertex and an incident hyperedge;
4. Each port of a vertex corresponds to exactly one vertical segment, either a segment of $sU(e)$ or a segment of $sL(e)$. The number of ports of a vertex is equal to the number of incident hyperedges for this vertex;
5. For every two hyperedges e_1 and e_2, it holds that e_1 and e_2 intersect if and only if a horizontal segment $sH(e_1)$ has crossing points with a vertical segment, a segment of $sU(e_2)$, or a segment of $sL(e_2)$, or vice versa. Note that this requirement means that the overlapping of segments becomes non-feasible for hypergraph drawing. Two examples of such non-feasible overlapping are demonstrated in Figure 6;
6. The number of hyperedge crossings is to be minimized.

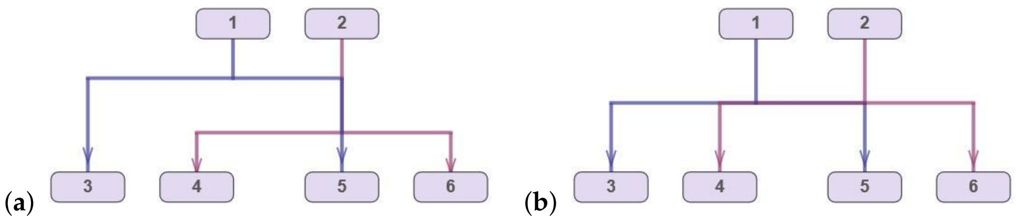

Figure 6. Examples of the non-feasible overlapping of hyperedges: (**a**) non-feasible overlapping of vertical segments; (**b**) non-feasible overlapping of horizontal segments.

3. Mathematical Model for Drawing a Hierarchical Hypergraph with Orthogonal Hyperedges and Ports

If hypergraph H contains hyperedges connecting the vertices of non-adjacent layers, then every such "long" hyperedge e has to be transformed into several hyperedges that connect the vertices of adjacent layers only, with additional (dummy) vertices being introduced for every e in the following way:

1. Find adjacent vertices u and v such that $\{u, v\} = \mathrm{argmax}_{u \in S, v \in T}(\lambda(u) - \lambda(v))$;
2. Insert a single dummy node on each layer l of hypergraph H, $\lambda(u) < l < \lambda(v)$;
3. Replace the long hyperedge e with $\lambda(v) - \lambda(u)$ hyperedges that connect the inserted dummy vertices with the incident vertices of the original long hyperedge e so that each new hyperedge connects the vertices of the adjacent layers. This step should hold all of the connections between nodes on adjacent layers that are defined by hyperedge e.

Consider the hypergraph $H' = (V', E_{H'}, \lambda')$, which is constructed from the hypergraph H (after inserting dummy vertices into it) by splitting each vertex into subsets of port vertices. For each vertex $u \in V$ (so-called "generating vertex"), sets K_U and K_L of the port vertices are introduced, $K_U(u) = \{u_U^1, u_U^2, \ldots, u_U^{m_U}\}$ and $K_L(u) = \{u_L^1, u_L^2, \ldots, u_L^{m_L}\}$, where m_U and m_L denote the numbers of ports of vertex u for the upper vertical segments and for the lower vertical segments of the incident hyperedges, respectively. Each port vertex in the set $K_U(u) \cup K_L(u)$ is considered a vertex of hypergraph H'. A function C is introduced as well. For each port vertex, the function C yields its generating vertex. Note that $\lambda'(v) = \lambda\,(u)\ \forall\ v \in K_U(u) \cup K_L(u)$. Figure 7 demonstrates an example of such a transformation of hypergraph H into hypergraph H'.

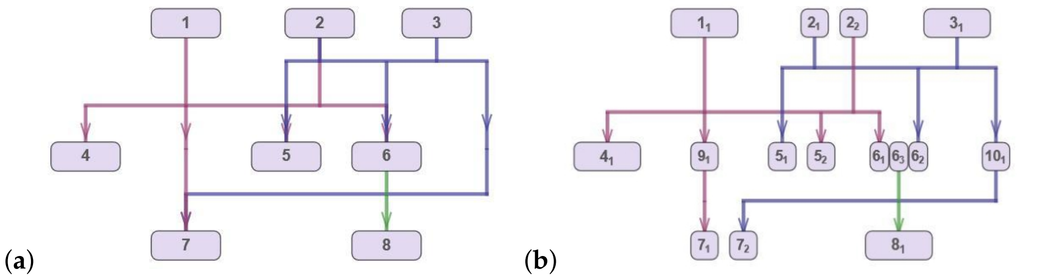

Figure 7. Transformation of hypergraph H into hypergraph H': (**a**) original hypergraph H; (**b**) hypergraph H' obtained after inserting dummy vertices and port vertices.

The following specific constraints are to be taken into account for the drawing of hypergraph H': port vertices with the same value for function C (those generated by one generating vertex) should be neighboring ones, i.e., between any two port vertices with the same value for function C, on every layer, there is no port vertex with another value for function C.

For the mathematical formulation of the hypergraph drawing problem, three sets of real variables are introduced—X, LMX, and RMX—as well as four sets of integer variables—AC, HO, UC, and LC.

Variable X_v denotes the x-coordinate of the vertex $v \in V'$. The variables LMX_e and RMX_e are the x-coordinates of the leftmost and rightmost nodes of hyperedge e, respectively.

A minimum distance $\rho(u,v)$ is known for each pair of port vertices $u,v \in V'$: $u,v \in K_U(C(u))$ (or $u,v \in K_L(C(u))$), $u \prec v$, or each pair of generating vertices $u,v \in V$: $u \prec v$. The order relation $u \prec v$ for a pair of vertices u and v means that they are in lexicographic order on the same layer.

A maximum distance $\delta(u',v')$ is known between each pair of port vertices $u',v' \in K_U(u)$ (or $u',v' \in K_L(u)$) of a generating vertex u, $u' \prec v'$.

A minimum distance $\theta(e_1,e_2)$ is known between vertical segments $sU(e_1)$ and $sL(e_2)$ or $sL(e_1)$ and $sU(e_2)$ for each pair of hyperedges $e_1,e_2 \in E_{H'}$: $e_1 \prec e_2$. The order relation $e_1 \prec e_2$ for a pair of hyperedges e_1 and e_2 means that they are in lexicographic order, and their source vertices are on the same layer.

Each hyperedge has a so-called "area of crossings". This can be defined as a set of points with coordinates (x,y) such that $x \in \left[\min_{v\in S\cup T} x(v), \max_{v\in S\cup T} x(v)\right]$ and y is between the ordinates of the source and target nodes. Figure 8 demonstrates the area of crossings of hyperedge e_1 with the incident vertices $(1,5,6)$. Vertices $1,2,4,6$, which are incident to another hyperedge e_2, are inside this region, and vertices 3 and 6 are outside of it.

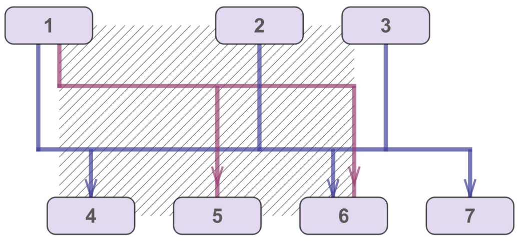

Figure 8. The area of crossings of hyperedge e_1.

For each pair of hyperedges (e_1, e_2) with source vertices on the same layers, where $e_1 = (S_1,T_1)$, $e_2 = (S_2,T_2)$ and $e_1 \prec e_2$, the integer variables AC and HO and dummy variables UC and LC are introduced as follows:

$$AC_{e_1,u} = \begin{cases} 0, & \text{if } u \in S_2 \cup T_2 \text{ is outside the area} \\ & \text{of crossings of hyperedge } e_1\,, \\ 1, & \text{otherwise.} \end{cases} \quad (1)$$

$$HO_{e_1,e_2} = \begin{cases} 0, & \text{if the horizontal segment of } e_2 \\ & \text{is higher than that of } e_1\,, \\ 1, & \text{otherwise.} \end{cases} \tag{2}$$

$$UC_{e_1,e_2} = \begin{cases} \sum\limits_{u \in S_2} AC_{e_1,u}, & \text{if the horizontal segment of } e_1 \\ & \text{is higher than that of } e_2\,, \\ \sum\limits_{u \in S_1} AC_{e_2,u}, & \text{otherwise.} \end{cases} \tag{3}$$

$$LC_{e_1,e_2} = \begin{cases} \sum\limits_{u \in T_1} AC_{e_2,u}, & \text{if the horizontal segment of } e_1 \\ & \text{is higher than that of } e_2\,, \\ \sum\limits_{u \in T_2} AC_{e_1,u}, & \text{otherwise.} \end{cases} \tag{4}$$

The variable UC_{e_1,e_2} is the number of crossings between the horizontal segment of one hyperedge of a corresponding pair of hyperedges and the upper segments of the other hyperedge. Similarly, the variable LC_{e_1,e_2} is the number of crossings between the horizontal segment of one hyperedge of this pair and the lower segments of the other hyperedge. The number of crossings for a pair of hyperedges e_1 and e_2 depends on the number of port vertices inside the area of crossings of these hyperedges and the relative vertical positions of their horizontal segments.

The following optimization criteria are applied in the model:

- Reducing the number of hyperedge crossings;
- Symmetric arrangement of source nodes relative to target nodes.

These can be formulated in the form:

$$\min \sum_{e_1,e_2 \in E_{H'}:\, e_1 \prec e_2} \Big(UC_{e_1,e_2} + LC_{e_1,e_2} \Big)\,, \tag{5}$$

$$\min \sum_{e=(S,T) \in E_{H'}} \sum_{u \in S} \left| X_u - \frac{1}{|T|} \sum_{v \in T} X_v \right|. \tag{6}$$

Note that, in all of the formulae below, $|\cdot|$ denotes the set cardinality for sets and an absolute value for numerical expressions.

The following set of constraints (7)–(10) is imposed for the generating and port vertices.

To ensure that no vertices overlap in the hypergraph layout, the distance between every pair of generating vertices u and v on the same layer ($u,\, v \in V :\ u \prec v$) should be equal to or greater than $\rho(u,v)$, i.e., $|X_u - X_v| \geq \rho(u,v)$. Taking into account that the abscissa of a generating vertex u can be calculated as the mean value of its port vertices with $K_U(u)$ abscissas, we re-write this condition in the form

$$\left| \frac{1}{|K_U(u)|} \sum_{u' \in K_U(u)} X_{u'} - \frac{1}{|K_U(v)|} \sum_{v' \in K_U(v)} X_{v'} \right| \geq \rho(u,v) \qquad \forall\, u,\, v \in V :\ u \prec v\,. \tag{7}$$

Analogous conditions are to be imposed for the distance between each pair of port vertices $u', v' \in K_U(u)$ (or $u', v' \in K_L(u)$) of every generating vertex $u \in V$. Note that they set both the minimum and the maximum distance between u' and v' as follows:

$$\rho(u',v') \leq |X_{u'} - X_{v'}| \leq \delta(u',v') \qquad \forall\, u \in V,\, \forall\, u',\, v' \in K_U(u) \text{ or } \forall\, u',\, v' \in K_L(u) :\ u' \prec v'\,. \tag{8}$$

The next set of conditions ensure that the mean value of the x-coordinates of the port vertices in the sets $K_U(u)$ and $K_L(u)$ are equal such that all of the port vertices in $K_U(u)$ and $K_L(u)$ are to be merged back into one generating vertex in the hypergraph drawing:

$$\left|\frac{1}{|K_U(u)|}\sum_{u'\in K_U(u)} X_{u'}\right| = \left|\frac{1}{|K_L(u)|}\sum_{v'\in K_L(u)} X_{v'}\right| \quad \forall\, u \in V:\ K_U(u) \neq \emptyset,\ K_L(u) \neq \emptyset. \tag{9}$$

The obvious condition that every port vertex u incident to hyperedge e lies between the leftmost and the rightmost vertices incident to hyperedge e defines the following connection between variables X_u, LMX_e, and RMX_e for every hyperedge e:

$$LMX_e \leq X_u \leq RMX_e \quad \forall\, e = (S,T) \in\ E_{H'},\ \forall\, u \in S \cup T\,. \tag{10}$$

A set of constraints (11)–(17) is formulated for each pair of hyperedges $e_1 = (S_1, T_1)$, $e_2 = (S_2, T_2)$ with source nodes on the same layer.

If the abscissa of a port vertex u incident to the hyperedge e_2 belongs to the interval that is defined by the x-coordinates of the leftmost and the rightmost vertices incident to hyperedge e_1, then the vertex u is inside the area of crossings of hyperedge e_1. In this case, there might be a crossing of a vertical segment of hyperedge e_2 connected with the vertex u and a horizontal segment of hyperedge e_1. The fact that a vertex is inside the area of crossings for a pair of hyperedges depends on the abscissa of the vertex and the values of the variables LMX_{e_1} and RMX_{e_1} in the following manner:

$$RMX_{e_1} - M \times AC_{e_1,u} \leq X_u \quad \text{or} \quad X_u \leq LMX_{e_1} + M \times AC_{e_1,u} \quad \forall\, e_1 = (S_1,T_1),\ e_2 = (S_2,T_2) \in E_{H'},\ \forall\, u \in S_2 \cup T_2\,, \tag{11}$$

Note that in expression (11) and in all of the formulae below, M is an arbitrary large positive number, $M \gg 1$.

The nonlinear conditions (3) and (4) are to be linearized in a standard manner as follows:

$$UC_{e_1,e_2} \geq \sum_{u\in S_2} AC_{e_1,u} - M \times (1 - HO_{e_1,e_2}) \quad \forall\, e_1,\ e_2 \in E_{H'}:\ e_2 = (S_2,T_2),\ e_1 \prec e_2\,. \tag{12}$$

$$LC_{e_1,e_2} \geq \sum_{u\in T_1} AC_{e_2,u} - M \times (1 - HO_{e_1,e_2}) \quad \forall\, e_1,\ e_2 \in E_{H'}:\ e_1 = (S_1,T_1),\ e_1 \prec e_2\,. \tag{13}$$

$$UC_{e_1,e_2} \geq \sum_{u\in S_1} AC_{e_2,u} - M \times HO_{e_1,e_2} \quad \forall\, e_1,\ e_2 \in E_{H'}:\ e_1 = (S_1,T_1),\ e_1 \prec e_2\,. \tag{14}$$

$$LC_{e_1,e_2} \geq \sum_{u\in T_2} AC_{e_2,u} - M \times HO_{e_1,e_2} \quad \forall\, e_1,\ e_2 \in E_{H'}:\ e_2 = (S_2,T_2),\ e_1 \prec e_2\,. \tag{15}$$

The vertical segments of every two hyperedges have no common point in the hypergraph layout. Hence, one has to ensure a minimum distance between the horizontal positions of the port vertices connected to these segments. The following conditions ensure the absence of non-feasible overlaps of hyperedge segments:

$$\begin{aligned} &\left|X_u - X_v\right| \geq M \times HO_{e_1,e_2} + \theta(e_1,e_2) \\ &\qquad \forall\, e_1 = (S_1,T_1),\ e_2 = (S_2,T_2) \in E_{H'} : e_1 \prec e_2 \\ &\qquad\qquad \forall\, u \in S_1\,, \quad \forall\, v \in T_2\,. \end{aligned} \tag{16}$$

$$\begin{aligned} &\left|X_u - X_v\right| \geq M \times (1 - HO_{e_1,e_2}) + \theta(e_1,e_2) \\ &\qquad \forall\, e_1 = (S_1,T_1), e_2 = (S_2,T_2) \in E_{H'} : e_1 \prec e_2, \\ &\qquad\qquad \forall\, u \in T_1\,, \quad \forall\, v \in S_2\,. \end{aligned} \tag{17}$$

The transitivity condition is to be satisfied for the positions of the horizontal segments of any triplet of hyperedges:

$$\begin{aligned} &0 \leq HO_{e_1,e_2} - HO_{e_1,e_3} + HO_{e_2,e_3} \leq 1 \\ &\qquad \forall\, e_1,\ e_2,\ e_3 \in E_{H'} : e_1 \prec e_2 \prec e_3\,. \end{aligned} \tag{18}$$

Finally, we determine the domains for all of the variables in the problem:

$$X_u \geq 0 \quad \forall\, u \in V' \quad \text{and} \quad LMX_e,\ RMX_e \in \mathbb{R} \quad \forall\, e \in E_{H'}\,. \tag{19}$$

$$HO_{e_1,e_2},\ UC_{e_1,e_2},\ LC_{e_1,e_2} \in \{0,\ 1\} \quad \forall\, e_1,\ e_2 \in E_{H'} : e_1 \prec e_2\,. \tag{20}$$

$$AC_{e,u} \in \{0,\ 1\} \quad \forall\, e = S, T \in E_{H'},\ \forall\, u \in V' : \exists\, v \in S \cup T,\ \lambda'(u) = \lambda'(v)\,. \tag{21}$$

The standard linearization procedure for nonlinear expressions (6)–(9), (11), (16), and (17) allows one to obtain a multi-objective mixed-integer program (5)–(21).

Figure 9 illustrates an example of the optimal layout of a hypergraph that was obtained on the basis of the model discussed above.

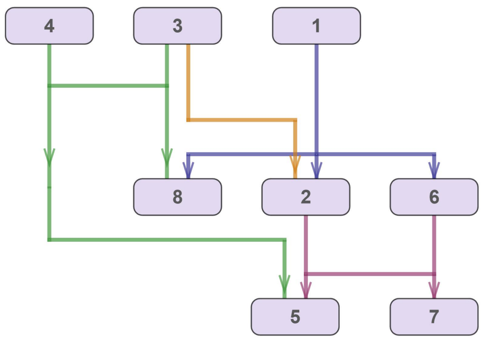

Figure 9. An example of the optimal layout of a hypergraph.

Two additional post-processing steps are required for an optimal hypergraph layout: first, all dummy vertices must be rendered as points, and second, all subsets of the port vertices corresponding to a generated vertex must be merged back into one vertex.

4. Minimizing the Height of the Hypergraph Drawing

The optimization model in (5)–(21) yields the relative order of the horizontal segments of the hyperedges. Clearly, some horizontal segments can be placed on the same horizontal level of the hypergraph layout without non-feasible overlapping. The smaller the number of horizontal levels where the horizontal segments are placed, the smaller the height of the hypergraph drawing will be.

Some modifications of the model in (5)–(21) are considered to minimize the number of horizontal levels for the horizontal segments of hyperedges.

First, two types of additional variables are introduced: real variables Y and binary variables SH.

The variables SH_{e_1,e_2} are defined for each pair of hyperedges $e_1 = (S_1, T_1)$ and $e_2 = (S_2, T_2)$ so that $e_1 \prec e_2$ in the following way:

$$SH_{e_1,e_2} = \begin{cases} 0, & \text{if the horizontal segments of } e_1 \text{ and } e_2 \text{ are on the same level,} \\ 1, & \text{otherwise.} \end{cases} \quad (22)$$

The variables Y_e determine the relative level for the horizontal segment of every hyperedge $e \in E_{H'}$. The smaller the value of Y_e is, the higher the horizontal segment of hyperedge e will be in the hypergraph drawing.

Second, condition (18) is replaced with the following system of constraints:

$$|Y_{e_1} - Y_{e_2}| \geq SH_{e_1,e_2} \qquad \forall\, e_1,\, e_2 \in E_{H'} :\ e_1 \prec e_2. \quad (23)$$

$$SH_{e_1,e_2} \geq \frac{1}{M}\left(\sum_{u\in\, S_2\cup T_2} AC_{e_1,u} + \sum_{v\in\, S_1\cup T_1} AC_{e_2,v}\right) \quad (24)$$
$$\forall\, e_1 = (S_1, T_1), e_2 = (S_2, T_2) \in E_{H'} : e_1 \prec e_2.$$

Conditions (23) and (24) are formulated for each pair of hyperedges e_1 and e_2 with source nodes on one layer, and they ensure that the values of Y_{e_1} and Y_{e_2} differ by a positive integer if the corresponding horizontal segments are on different levels. Again, a constant $M \gg 1$ is used.

Finally, the following criterion is added to the optimization model in (5)–(21), (23), and (24):

$$\min \sum_{e\in E_{H'}} Y_e. \quad (25)$$

Figure 10 shows an example of the optimal layout of the hypergraph obtained using the described variant of the model. The horizontal segments of some hyperedges have the same ordinate, which makes it possible to reduce the height of the figure.

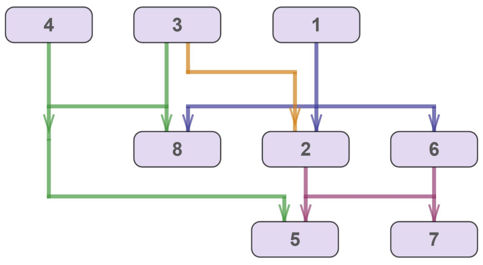

Figure 10. An example of the optimal layout of the hypergraph with the same ordinates of horizontal segments.

5. A Mathematical Heuristic for the Hypergraph Layout

The problem of the hypergraph layout in (5)–(21) can be solved using the following five-step mathematical heuristic:

1. Find the order of the vertices on each layer;
2. Assign preliminary abscissas to the vertices;
3. Find the relative order of horizontal segments for orthogonal hyperedges with source nodes on the same layer;
4. Remove the non-feasible overlapping of vertical segments;
5. Assign ports to the vertical segments.

Note that the aesthetic criteria used in the heuristic coincide with those in Section 3. The mathematical heuristic is sketched below; see Algorithm 1.

Algorithm 1: A mathematical heuristic for the hypergraph layout.

```
Data: hierarchical hypergraph $H = (V, E_H, \lambda)$
Result: hypergraph drawing $D$
$ord \leftarrow ordering(hypergraphRestructuring_1(H))$;
$X \leftarrow assigningAbscissas(hypergraphRestructuring_2(H), ord)$;
$HO \leftarrow horizontalSegmentsOrdering(H, X)$;              /* (26)-(30) */
$\widetilde{X} \leftarrow removeOverlapping(H, X, HO)$;
$D \leftarrow assigningPorts(H, \widetilde{X}, HO)$;            /* (31)-(33) */
```

The first step (vertex ordering) includes a pre-processing procedure for the initial hypergraph H; the corresponding function is denoted as *hypergraphRestructuring*$_1$ in Algorithm 1. A detailed description of this function that returns a graph corresponding to hypergraph H is given in Algorithm 2.

Algorithm 2: The *hypergraphRestructuring*$_1$ function.

```
Data: hierarchical hypergraph $H = (V, E_H, \lambda)$
Result: hierarchical graph $G_1$
$i \leftarrow |V|$;
$E_{G_1} \leftarrow \emptyset$;
$V_{G_1} \leftarrow V$;
forall $e = (S, T) \in E_H$ do
    $V_{G_1} \leftarrow V_{G_1} \cup \{v_{i+1}\}$;
    forall $u \in S$ do
        $E_{G_1} \leftarrow E_{G_1} \cup (u, v_{i+1})$;
        $\lambda_{G_1}(u) \leftarrow 2 \times \lambda(u) - 1$;
    end
    forall $u \in T$ do
        $E_{G_1} \leftarrow E_{G_1} \cup (v_{i+1}, u)$;
        $\lambda_{G_1}(u) \leftarrow 2 \times \lambda(u) - 1$;
    end
    Choose $u \in S$;
    $\lambda_{G_1}(v_{i+1}) \leftarrow \lambda_{G_1}(u) + 1$;
    $i \leftarrow i + 1$;
end
$G_1 \leftarrow (V_{G_1}, E_{G_1}, \lambda_{G_1})$;
```

To convert the hypergraph into a graph with the *hypergraphRestructuring*$_1$ function, the so-called "balancing vertex" is introduced for each hyperedge $e = (S, T)$; see Figure 11a,b. Then, each vertex $v \in S \cup T$ incident to the hyperedge is connected by a directed edge to the balancing vertex. This approach enables one to correctly take into account all possible intersections of edges.

The order of the vertices on the layers from *hypergraphRestructuring*$_1(H)$ is determined using the *ordering* function; see Algorithm 1. The vertex-ordering problem is solved using an integer program [10] that minimizes the number of edge crossings in a directed acyclic graph.

To assign preliminary abscissas to the vertices (step 2 of the mathematical heuristic), the function *assigningAbscissas* is used. Again, hypergraph H is pre-processed with the *hypergraphRestructuring*$_2$ function; see Algorithm 3. As a result of the transformation, every pair of adjacent vertices on the adjacent layers is connected by a directed edge; see Figure 11c.

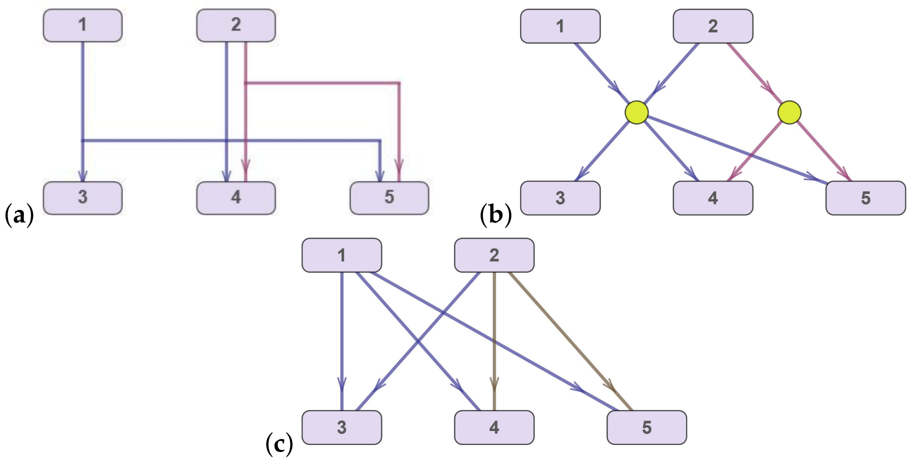

Figure 11. An example of a hypergraph transformation: (**a**) original hypergraph H; (**b**) restructuring a hypergraph into a directed graph with balancing vertices; (**c**) restructuring a hypergraph into a directed graph.

Algorithm 3: The *hypergraphRestructuring*$_2$ function.

```
Data: hierarchical hypergraph $H = (V, E_H, \lambda)$
Result: hierarchical graph $G_2$
$E_{G_2} \leftarrow \emptyset$;
forall $e = (S, T) \in E_H$ do
    forall $u \in S$ do
        forall $v \in T$ do
            if $(u, v) \notin E_{G_2}$ then
                $E_{G_2} \leftarrow E_{G_2} \cup (u, v)$;
            end
        end
    end
end
$G_2 \leftarrow (V, E_{G_2}, \lambda)$;
```

Then, x-coordinates are assigned to the vertices based on the solution to a linear programming problem with objective function (6); see [9]. Note that the relative vertex order on each layer remains.

Once step 2 is over, three introduced functions—*horizontalSegmentsOrdering*, *removeOverlapping*, and *assigningPorts*—are applied to obtain the hypergraph drawing. These functions correspond to steps 3, 4, and 5 of the mathematical heuristic, and their detailed description is presented below.

Knowing the x-coordinates of the vertices, the order of the horizontal segments of the hyperedges is determined. The objective function L that is minimized in step 3 is the number of hyperedge crossings:

$$\min \sum_{e_1, e_2 \in E_H} CT_{e_1,e_2}. \tag{26}$$

For each pair of hyperedges $e_1 = (S_1, T_1)$ and $e_2 = (S_2, T_2)$, one can calculate the number of source nodes $v \in S_2$ and target nodes $u \in T_2$ that are inside the area of crossings of hyperedge e_1. We denote these numbers as $acs(e_1, e_2)$ and $act(e_1, e_2)$, respectively. Then, the system of constraints for the integer programming problem with objective function (26) is as follows:

$$\begin{gathered} CT_{e_1,e_2} \geq acs(e_1, e_2) + act(e_2, e_1) - M \times (1 - HO_{e,e_2}) \\ \forall\ e_1,\ e_2 \in E_{H'} : e_1 \prec e_2. \end{gathered} \tag{27}$$

$$CT_{e_1,e_2} \geq acs(e_2,e_1) + act(e_1,e_2) - M \times HO_{e_1,e_2} \quad \forall\, e_1,\, e_2 \in E_{H'} : e_1 \prec e. \tag{28}$$

$$0 \leq HO_{e_1,e_2} - HO_{e_1,e_3} + HO_{e_2,e_3} \leq 1 \quad \forall\, e_1,\, e_2,\, e_3 \in E_{H'} :\; e_1 \prec e_2 \prec e_3. \tag{29}$$

$$CT_{e_1,e_2} \geq 0,\; HO_{e_1,e_2} \in \{0,\, 1\} \quad \forall\, e_1,\, e_2 \in E_{H'} : e_1 \prec e_2\,. \tag{30}$$

Constraints (27) and (28) are a simplified version of conditions (12)–(15) from the optimization model described in Section 3. The simplification is based on the fact that the x-coordinates of vertices are known. Expression (29) coincides with condition (18) and ensures that the transitivity condition is satisfied for the positions of the horizontal segments of any triplet of hyperedges. Condition (30) determines the domains of all of the variables in the problem.

The function *horizontalSegmentsOrdering*—see Algorithm 1—is the solution to the mixed-integer program in (26)–(30). It returns the relative order of horizontal segments of the hyperedges.

In step 4, all non-feasible overlaps of vertical segments of hyperedges are removed by using the integer program proposed in [7]. This solution is denoted as the *removeOverlapping* function in Algorithm 1. Non-feasible overlaps must be eliminated by shifting the vertices on layers by a certain integer number of “single shifts” for each corresponding vertex x-coordinate while maintaining the relative order of the vertices on the layers. The objective function of the integer programming problem minimizes the total number of shifts of vertices, and the constraints are (7)–(21) with fixed values of the HO variables for every pair of hyperedges.

The last step of the mathematical heuristic is the port assignment procedure when the relative order of ports is determined by the *assigningPorts* function for each hypergraph vertex—see Algorithm 4—where the *arrangingSymmetrically* function determines the x-coordinates of the port vertices by placing them symmetrically to the x-coordinate of the generating vertex without breaking the relative order of the port vertices.

Algorithm 4: The *assigningPorts* function.

```
Data: hierarchical hypergraph $H = (V, E_H, \lambda)$, abcissas' vertices $\widetilde{X}$, horizontal
      segment ordering $HO$
Result: hypergraph drawing $D$
forall $u \in V$ do
    forall $K \in \{K_U(u), K_L(u)\}$ do
        forall $p \in K$ do
            $s(p) \leftarrow 0$;
            forall $j \in K \setminus \{p\} : j \prec p$ do
                $s(p) \leftarrow s(p) + \sum_j ORD_{j,p}$;                      /* (33) */
            end
            forall $i \in K \setminus \{p\} : p \prec i$ do
                $s(p) \leftarrow s(p) + \sum_j (1 - ORD_{p,i})$;                /* (33) */
            end
        end
        Sort $K$ in increasing order by the value of $s(p)$;
        $arrangingSymmetrically(K, \widetilde{X}(u))$;
    end
    $D \leftarrow$ Draw $H$ as far as all of the necessary characteristics of the hypergraph
      elements are known.
end
```

For every generating vertex u, binary variables $ORD_{u',v'}$ are introduced for each pair of port vertices $u', v' \in K_U(u)$ or $u', v' \in K_L(u)$ (relation $u' \prec v'$ holds) as follows:

$$ORD_{u',v'} = \begin{cases} 0, & \text{if port vertex } u' \text{ is to the right of port vertex } v', \\ 1, & \text{otherwise.} \end{cases} \tag{31}$$

A numerical parameter $pos(u, e)$ is defined for each pair consisting of a hyperedge $e = (S, T) \in E_{H'}$ and a generating vertex $u \in S \cup T$:

$$pos(u,e) = \begin{cases} 2, & \text{if } X_u = \max\limits_{v \in S \cup T} X_v\,, \\ 0, & \text{if } X_u = \min\limits_{v \in S \cup T} X_v\,, \\ 1, & \text{otherwise.} \end{cases} \tag{32}$$

Consider a pair of a hyperedge $e = (S, T) \in E_H$ and a generating vertex $u \in S \cup T$. If $pos(u, e) = 0$, then u is the leftmost vertex in a set of vertices incident to e, and if $pos(u, e) = 2$, then u is the rightmost one. The equality $pos(u, e) = 1$ means that vertex u is between the leftmost and the rightmost vertices incident to e. Parameter $pos(u, e)$ enables one to define the value of $ORD_{u,v}$, i.e., the relative order of two port vertices u and v, as follows:

$$ORD_{u,v} = \begin{cases} \mathcal{U}(pos(C(u), e_1) - pos(C(v), e_2)), & \text{if } pos(C(u), e_1) \neq pos(C(v), e_2), \\ 1 - HO_{e_1,e_2}, & \text{if } pos(C(u), e_1) = type(u), \\ HO_{e_1,e_2}, & \text{else.} \end{cases} \tag{33}$$

where $\mathcal{U}$ denotes the unit step function, u and v are incident to the hyperedges e_1 and e_2, respectively, $u, v \in K_U(C(u))$ or $u, v \in K_L(C(u))$, $u \prec v$, and a numerical parameter $type(u)$ takes the value 2 if port vertex u for the upper vertical segment of $sU(e_1)$ or 1 in the other case.

Once the relative order of all pairs of port vertices u', v', $C(u') = C(v')$ is known, all of the port vertices of the generating vertex u are positioned so that their mean x-coordinate coincides with that of the center of u. The minimal and maximal distances $\rho(u', v')$ and $\delta(u', v')$ (see Section 3) should be taken into account. Figure 12 illustrates a hypergraph layout that was obtained using the mathematical heuristic approach described above.

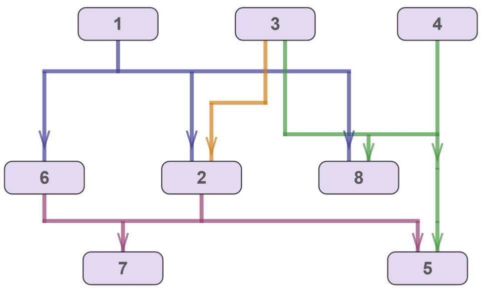

Figure 12. An example of layout of a hypergraph obtained with the mathematical heuristic.

6. Numerical Results and Discussion

The approach to a solution of the hypergraph drawing problem proposed above was verified using synthetic initial data. The corresponding numerical results were analyzed for the multi-objective mixed-integer program in (5)–(21) and the mathematical heuristic (see Section 5) for a set of generated hypergraphs.

Wolfram Mathematica 12.3 [15] was used as a programming tool together with Gurobi Optimizer 9.1 [16]. All the numerical calculations were performed on an HP desktop, with an Intel Core i5 1.60 GHz CPU, 8 GB of RAM, and the Windows 10 operating system. The numerical results are presented in Table 1.

Three approaches were applied to obtain the numerical results, namely, the mathematical heuristic (see Section 5), mixed-integer program (5)–(21) with a higher priority of the objective function (5), and the same mixed-integer program enhanced by an initial feasible solution derived with the mathematical heuristic approach.

Table 1. Numerical results obtained using the mathematical heuristic (MH) and mixed-integer program (MIP) for a set of generated hypergraphs.

ID	$(\lvert\mathbf{V}\rvert, \lvert\mathbf{E_H}\rvert)$	Time, in sec			Number of Crossings	
		MH	MIP	MH+MIP	MH	MIP/MH+MIP
ms_n8_he4	(8, 4)	0.5	1.6 (<1)	1.34 (1)	3	1
ms_n12_he6	(12, 6)	0.52	2.78 (2)	2.52 (2)	4	1
ms_n14_he7	(14, 7)	0.53	3.02 (2)	2.91 (2)	3	2
ms_n15_he7	(15, 7)	0.59	10.12 (5)	5.82 (4)	16	3
ms_n15_he8	(15, 8)	0.5	3.05 (2)	2.4 (2)	2	0
ms_n16_he9	(16, 9)	0.57	21.07 (14)	4.61 (4)	3	1
ms_n18_he10	(18, 10)	0.63	44.87 (24)	6.9 (6)	13	4
ms_n19_he10	(19, 10)	0.58	130.2 (91)	9.51 (8)	7	3
ms_n20_he11	(20, 11)	0.63	497.9 (406)	15.86 (14)	17	7
ms_n27_he11	(27, 11)	0.86	34.34 (12)	8.79 (8)	22	8
ms_n22_he14	(22, 14)	0.64	39.56 (33)	7.02 (6)	14	4
ms_n41_he19	(41, 19)	2.29	690.8 (411)	26.33 (24)	70	16
ss_n10_he7	(10, 7)	0.53	4.48 (4)	3.89 (3)	3	1
ss_n11_he7	(11, 7)	0.52	1.71 (<1)	1.69 (1)	4	0
ss_n20_he13	(20, 13)	0.57	46.81 (44)	6.25 (6)	2	0
ss_n27_he21	(27, 21)	0.75	35.81 (5)	7.84 (6)	6	1
ss_n40_he26	(40, 26)	1.43	136.4 (85)	11.85 (10)	25	7

Each hypergraph is described by its ID. If the ID contains the prefix `ms`, then the hypergraph is considered to have multi-source hyperedges; otherwise, all the hyperedges in the hypergraph are single-source ones (prefix `ss` in the ID). The numbers of vertices and hyperedges are indicated in the column $(\lvert\mathbf{V}\rvert, \lvert\mathbf{E_H}\rvert)$. The column "Time in sec" gives information on the computation time for each numerical experiment. There is additional information (in parentheses) on how long it takes for Gurobi to find a solution that cannot be optimized with further iterations.

7. Conclusions

A new mathematical model for drawing a hierarchical hypergraph with vertex ports and multi-source orthogonal hyperedges is discussed in this paper. The model is based on the formulation of a mixed-integer program. One of the advantages of the model is that it enables one to simultaneously take into account a set of optimization criteria, including aesthetics ones (for overall hypergraph drawing), as well as the minimization of hyperedge crossings. The numerical results demonstrate that the number of hyperedge crossings can be minimized compared to the mathematical heuristic approach.

One of the application areas of the model is in financial flow visualization in information systems to create an initial drawing of a hypergraph. The initial layout is corrected when actual data are updated using the dynamic hierarchical graph-drawing algorithm [9].

Further directions of research include:

- The generalization of the proposed mathematical model for reverse-directed hyperedges by adding port vertices to the model on the left and right sides of the generating vertices;
- The development of a fast and efficient algorithm (heuristic, metaheuristic, column generation, etc.) for drawing large-scale hypergraphs.

Author Contributions: Conceptualization, G.F. and Y.V.; methodology, Y.V.; software, V.P.; validation, V.P.; formal analysis, G.F.; investigation, G.F., Y.V. and V.P.; resources, V.R.; data curation, G.F.; writing—original draft preparation, V.P.; writing—review and editing, G.F. and Y.V.; visualization, V.P.; supervision, G.F.; project administration, V.R.; funding acquisition, V.R. All authors have read and agreed to the published version of the manuscript.

Funding: The research was partially funded by the Ministry of Science and Higher Education of the Russian Federation as part of the World-Class Research Center program: Advanced Digital Technologies (contract No. 075-15-2020-903 dated 16.11.2020).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Collection of the hypergraphs generated: https://drive.google.com/drive/folders/1GQL3gKQsVoDMrAt-YkHCYhaBabM7wBN- (Accessed: 14 December 2021). Additional data presented in this study are available on request from the authors.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Spönemann, M. *Graph Layout Support for Model-Driven Engineering*; BoD–Books on Demand: Norderstedt, Germany, 2015
2. Helmke, S.; Goetze, B.; Scheffler, R.; Wrobel, G. Interactive, Orthogonal Hyperedge Routing in Schematic Diagrams Assisted by Layout Automatisms. In *Diagrammatic Representation and Inference. Diagrams 2021*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2021. [CrossRef]
3. Schulze, C.D.; Spönemann, M.; von Hanxleden, R. Drawing layered graphs with port constraints. *J. Vis. Lang. Comput. Issue Diagr. Aesthet. Layout* **2014**, *25*, 89–106. [CrossRef]
4. Jünger, M.; Mutzel, P.; Spisla, C. More Compact Orthogonal Drawings by Allowing Additional Bends. *Information* **2018**, *9*, 153. [CrossRef]
5. Sugiyama, K.; Tagawa, S.; Toda, M. Methods for visual understanding of hierarchical system structures. *IEEE Trans. Syst. Man Cybern.* **1981**, *11*, 109–125. [CrossRef]
6. Sander, G. Layout of Directed Hypergraphs with Orthogonal Hyperedges. *Graph Draw.* **2003**, 381–386. [CrossRef]
7. Vasiliev, Y.M.; Fridman, G.M. Cooperation scheme visualization: Hyperedge routing method for hierarchical multilayer hypergraph. *Sovrem. Ekon. Probl. Resheniia* **2017**, *3*, 18–33. (In Russian) [CrossRef]
8. Junger, M.; Mutzel, P. 2-layer straightline crossing minimization: Performance of exact and heuristic algorithms. *J. Graph Algorithms Appl.* **1997**, 1. [CrossRef]
9. Ismaeel, A.A.K. Dynamic Hierarchical Graph Drawing. Ph.D. Thesis, Karlsruher Instituts fur Technologie (KIT), Karlsruhe, Germany, 2012.
10. Healy, P.; Nikolov, N.S. Hierarchical drawing algorithms. In *Handbook on Graph Drawing and Visualization*; CRC: Boca Raton, FL, USA, 2013; pp. 409–453.
11. Spönemann, M.; Schulze, C.D.; Rüegg, U.; von Hanxleden, R. Counting Crossings for Layered Hypergraphs. In *Diagrammatic Representation and Inference. Diagrams 2014*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; p. 8578. [CrossRef]
12. Eschbach, T.; Guenther, W.; Becker, B. Orthogonal hypergraph drawing for improved visibility. *J. Graph Algorithms Appl.* **2006**, *10*, 141–157. [CrossRef]
13. Šuvakov, M.; Andjelković, M.; Tadić, B. Hidden geometries in networks arising from cooperative self-assembly. *Sci. Rep.* **2018**, *8*, 1–10. [CrossRef]
14. Tadić, B.; Andjelković, M.; Šuvakov, M.; Rodgers, G.J. Magnetisation Processes in Geometrically Frustrated Spin Networks with Self-Assembled Cliques. *Entropy* **2020**, *22*, 336. [CrossRef] [PubMed]
15. Wolfram Research. Wolfram Mathematica. Available online: https://www.wolfram.com/mathematica/ (accessed on 14 December 2021).
16. Gurobi Optimization, LLC. Gurobi Optimizer. Available online: https://www.gurobi.com/products/gurobi-optimizer/ (accessed on 14 December 2021).